travisyeh_pset3-2

February 28, 2017

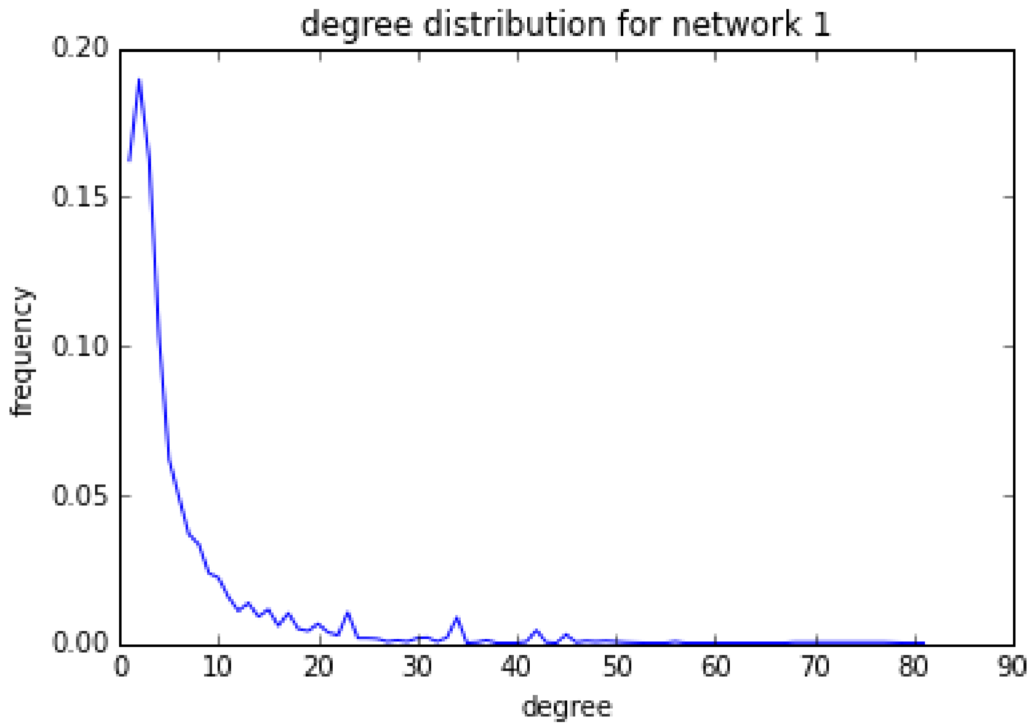
```
In [30]: import numpy as np
         import matplotlib.pyplot as plt
         import math
         import operator
         %matplotlib inline
```

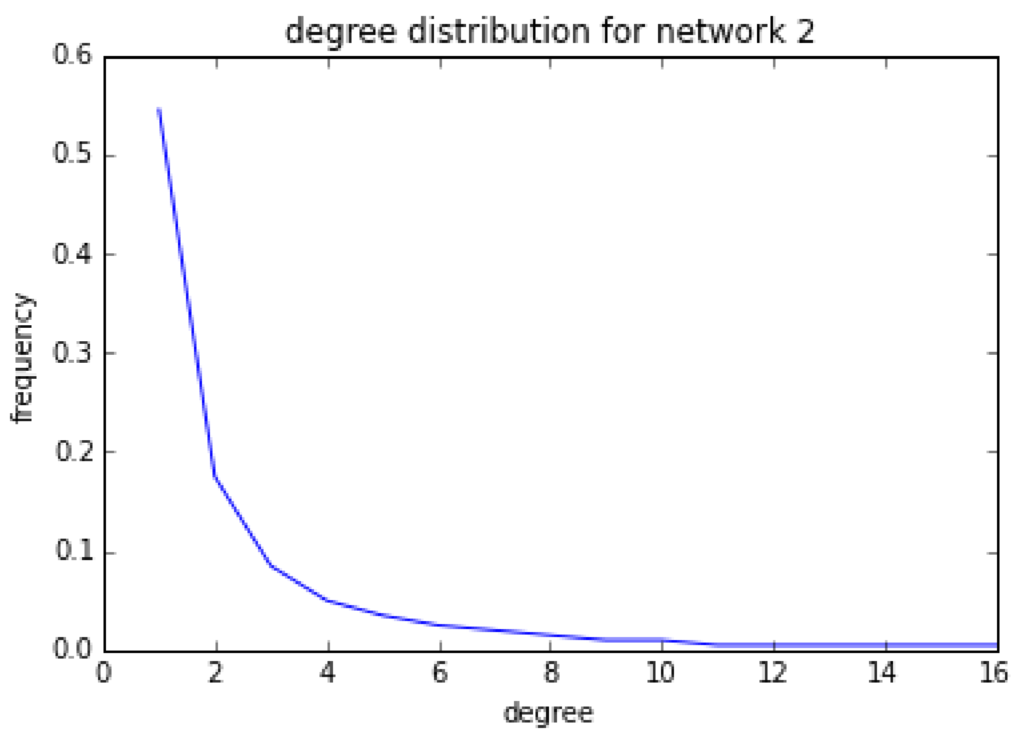
1 Problem 5a

```
In [9]: '''
        Function that reads in one of the network files and creates a list of the
        nodes and their degrees
        Input:
            - filename: the name of the file to be read
        Output:
            - result: a Python dict where the key represents the vertex and the va
                represents the degree of that node

        '''
        def readFile(filename):
            result = {}

            # open the file and read it in
            graph = open(filename, "r")

            for pair in graph:
                pair = pair.split()
                first = pair[0]

                # check to see if this is a vertex that already exists -- if not, w
                # note that since each pair is listed in both directions,
                if result.get(first):
                    result[first] += 1
                else:
                    result[first] = 1

            return result
```

In [10]:
```
'''
Function that finds the degree distribution for a graph. returns things as
the key is the degree and the value is the number of nodes with that degre

'''
def findDistribution(graph):
    result = {}
    # find the total number of nodes
    total = len(graph)

    # go through the given graph and keep track of how many nodes have the
    for node, degree in graph.items():
        if result.get(degree):
            result[degree] += 1
        else:
            result[degree] = 1

    # divide the sums by the total to find the distribution
    for degree in result:
        result[degree] = float(result[degree])/float(total)

    return result
```

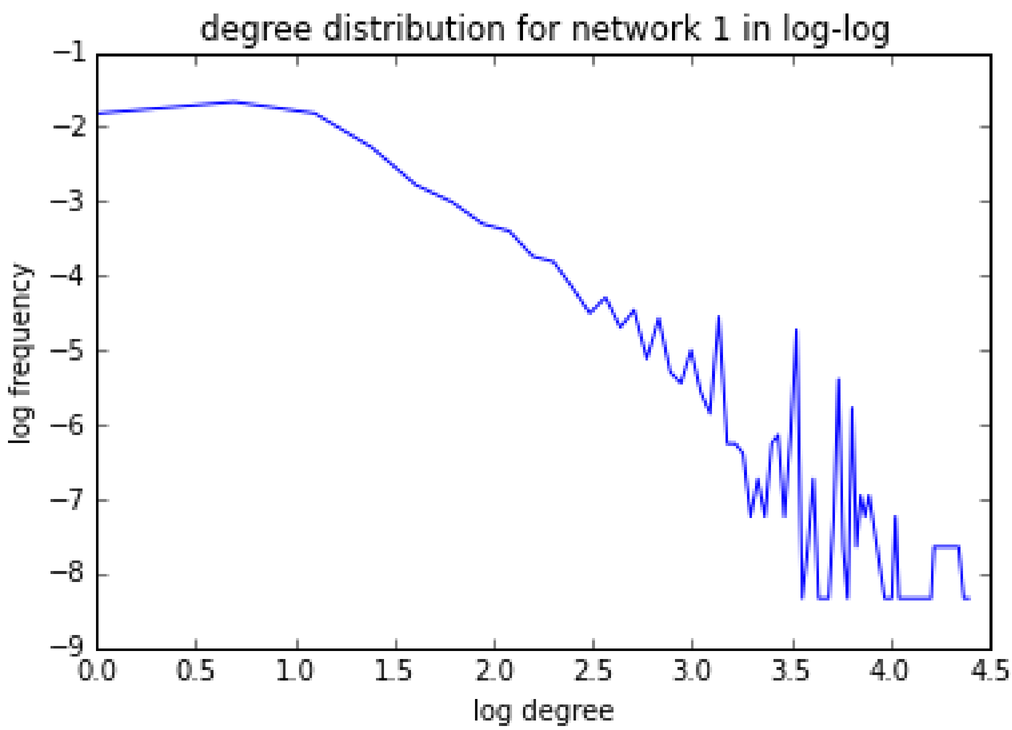
In [11]:
```
# load up the networks and make the plots
network1 = readFile("network1.txt")
network2 = readFile("network2.txt")

dist1 = findDistribution(network1)
dist2 = findDistribution(network2)

# plot the stuff for the first network
plt.plot(dist1.keys(), dist1.values())
plt.title("degree distribution for network 1")
plt.xlabel("degree")
plt.ylabel("frequency")

plt.show()

# now do the second network
plt.plot(dist2.keys(), dist2.values())
plt.title("degree distribution for network 2")
plt.xlabel("degree")
plt.ylabel("frequency")

plt.show()
```

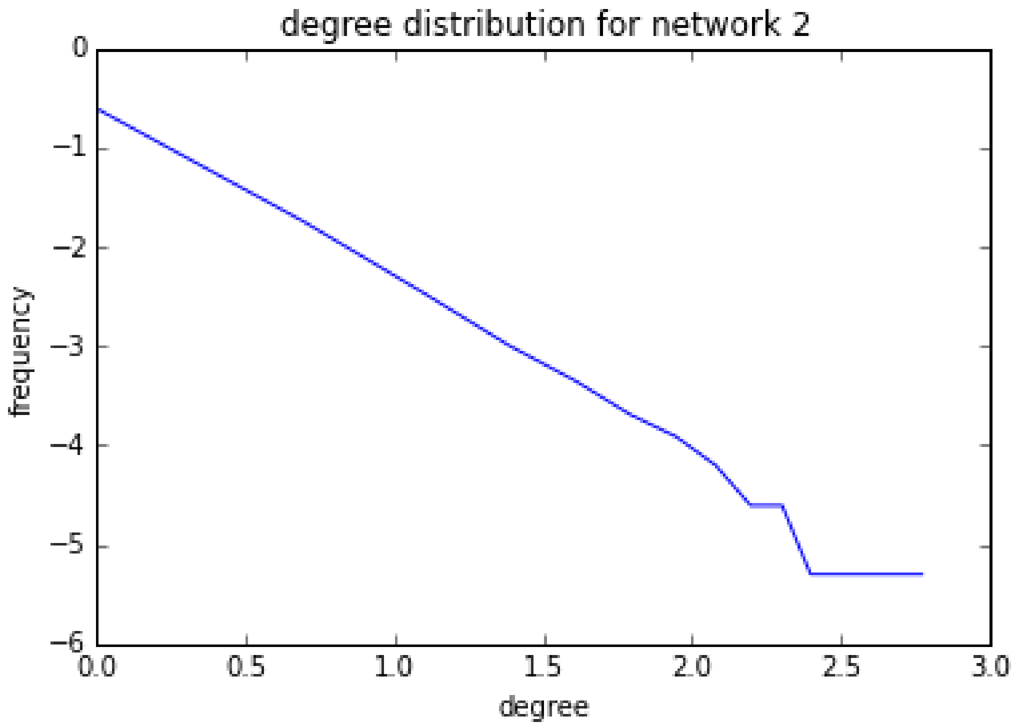

degree distribution for network 1
0.20
0.15
0.10
0.05
0.00
frequency
0
10
20
30
40
50
60
70
80
90
degree


degree distribution for network 2
0.6
0.5
0.4
0.3
0.2
0.1
0.0
frequency
0
2
4
6
8
10
12
14
16
degree

2 Problem 5b

In [102]:
```
# plot the stuff for the first network
plt.plot([math.log(i) for i in dist1.keys()], [math.log(i) for i in dist1
plt.title("degree distribution for network 1 in log-log")
plt.xlabel("log degree")
plt.ylabel("log frequency")

plt.show()

# now do the second network
plt.plot([math.log(i) for i in dist2.keys()], [math.log(i) for i in dist2
plt.title("degree distribution for network 2")
plt.xlabel("degree")
plt.ylabel("frequency")

plt.show()
```





Based on the graphs above, it seems that network 2 looks more linear, while network 1 displays some erratic behavior towards the larger end of log(degree).

3 Problem 5c

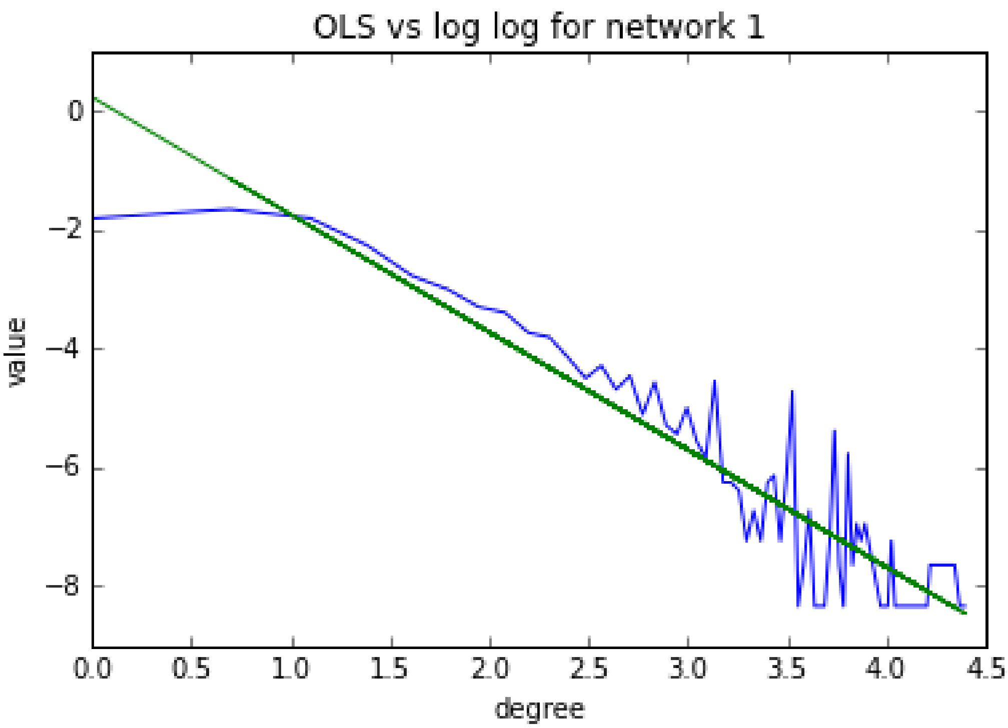
In [46]:
```
# create the log-log versions of the distributions
log_dist1 = {}
log_dist2 = {}

for k,v in dist1.items():
    log_dist1[math.log(k)] = math.log(v)

for k,v in dist2.items():
    log_dist2[math.log(k)] = math.log(v)
```

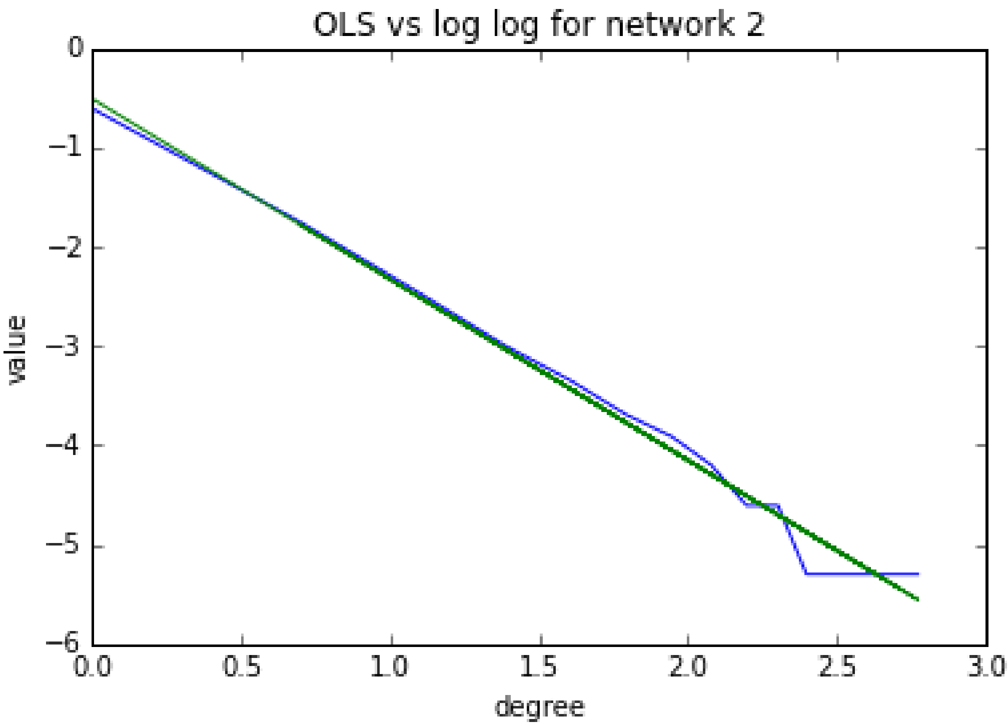
In [47]:
```
# fit the regression using numpy and plot the results
ols1 = np.polyfit(log_dist1.keys(), log_dist1.values(), 1)
ols2 = np.polyfit(log_dist2.keys(), log_dist2.values(), 1)
```

In [48]:
```
print "Coefficients for network 1:"
print ols1
print "Coefficients for network 2:"
print ols2
```

```
Coefficients for network 1:
[-1.97409568  0.22023497]
Coefficients for network 2:
[-1.82035669 -0.50814429]
```

```
In [57]: # plot the regressions
         plt.plot([math.log(i) for i in dist1.keys()], [math.log(i) for i in dist1.
         plt.plot(log_dist1.keys(), [ols1[0]*i + ols1[1] for i in log_dist1.keys()]
         plt.title("OLS vs log log for network 1")
         plt.xlabel("degree")
         plt.ylabel("value")
         plt.show()

         plt.plot([math.log(i) for i in dist2.keys()], [math.log(i) for i in dist2.
         plt.plot(log_dist2.keys(), [ols2[0]*i + ols2[1] for i in log_dist2.keys()]
         plt.title("OLS vs log log for network 2")
         plt.xlabel("degree")
         plt.ylabel("value")
         plt.show()
```





4 Problem 5d

```
In [66]: '''

         Function to find the MLE for alpha
         Input:
             - degrees: the array of degrees mentioned in the problem
         Output:
             - alpha: the MLE

         '''
         def findMLE(degrees):
             degree_min = min(degrees)
             total = len(degrees)
             alpha = 0.0

             # iterate through to do the summation in the formula from section note
             for i in degrees:
                 alpha += math.log(float(i)/float(degree_min))

             return 1 + total*(1.0/float(alpha))

In [68]: mle1 = findMLE(list(network1.values()))
```

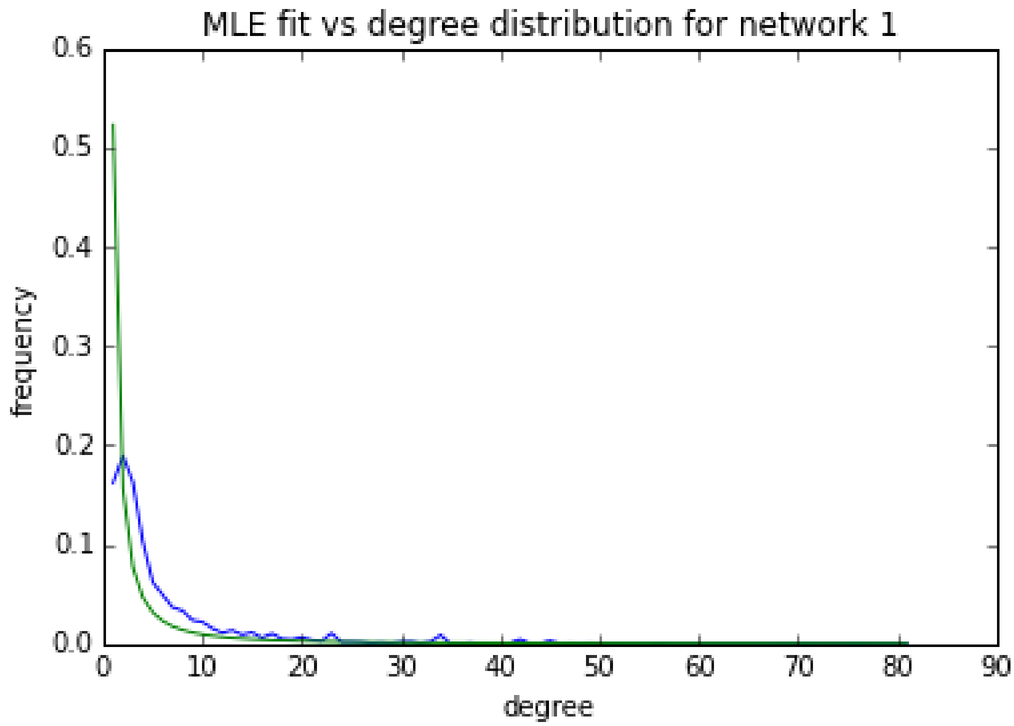
```
mle2 = findMLE(list(network2.values()))

print "MLE1: " + str(mle1)
print "MLE2: " + str(mle2)
```

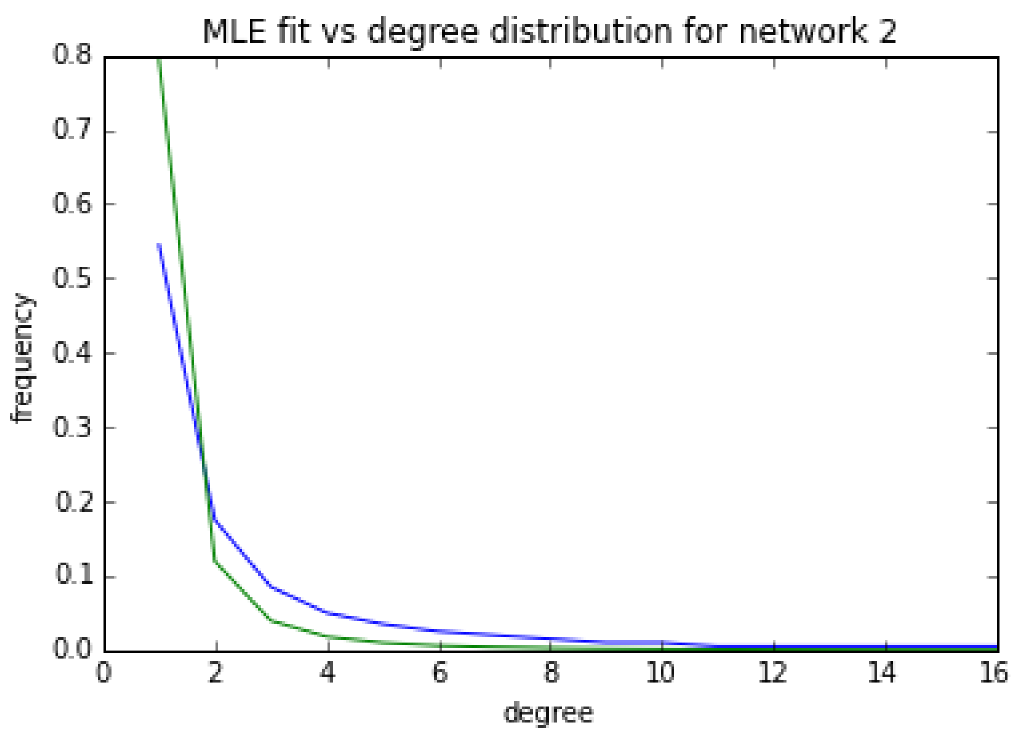
```
MLE1: 1.74219107727
MLE2: 2.72990330106
```

```
In [100]: # here, we find the constants for the power law
          c1 = 1.0/sum([x**-mle1 for x in dist1.keys()])
          c2 = 1.0/float(sum([x**-mle2 for x in dist2.keys()]))

          print "c1: " + str(c1)
          print "c2: " + str(c2)
```

```
c1: 0.522823402964
c2: 0.792888189417
```

```
In [101]: # plot the stuff for the first network
          plt.plot(dist1.keys(), dist1.values())
          plt.plot(dist1.keys(), [c1*(i**(-mle1)) for i in dist1.keys()])
          plt.title("MLE fit vs degree distribution for network 1")
          plt.xlabel("degree")
          plt.ylabel("frequency")

          plt.show()

          # now do the second network
          plt.plot(dist2.keys(), dist2.values())
          plt.plot(dist2.keys(), [c2*(i**(-mle2)) for i in dist2.keys()])
          plt.title("MLE fit vs degree distribution for network 2")
          plt.xlabel("degree")
          plt.ylabel("frequency")

          plt.show()
```


MLE fit vs degree distribution for network 1
frequency
degree
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0 10 20 30 40 50 60 70 80 90


MLE fit vs degree distribution for network 2
frequency
degree
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0 2 4 6 8 10 12 14 16

5 Problem 5e

See parts (c) and (d) for their respective plots (the regression lines are in green). It looks like the OLS method fit network 2 better than it did network 1, while the MLE method resulted in fits that looked about the same for both. The difference in how the OLS method fit might be a result of the degree distribution for network 1 being more erratic near the tail end, while the degree distribution for network 2 looks almost linear when put on a log-log model. Thus, while the OLS method suggests that only network 2 would be a good fit for the power law model, the MLE method takes care of the erraticness and shows that both might be a good fit after all.

```
In [ ]:
```